SYBASE®
An SAP® Company

Fundamentals

Sybase Unwired Platform 1.5.2

DOCUMENT ID: DC01204-01-0152-02
LAST REVISED: February 2011

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

CHAPTER 1 Introduction to Fundamentals

This document introduces you to fundamental mobile computing concepts, then identifies how you can use Sybase® Unwired Platform to implement the concepts in your enterprise.

Fundamentals is targeted towards users who are developing mobile device applications, but executives, system integrators, system implementers, and system administrators who need a high-level understanding of Sybase Unwired Platform may also find it useful.

Introduction to Sybase Unwired Platform

Unwired Platform provides an integrated solution for mobilizing your enterprise applications to multiple device platforms using the same infrastructure and development investment.

Unwired Platform covers end-to-end aspects of mobilizing an application—providing standards-based access to data sources, building heterogeneous device applications with reusable object-oriented frameworks, supporting end-to-end security, and managing deployment, devices, users, and applications.

This guide is to familiarize you with the platform, provide an overview of the platform capabilities, and show how the platform addresses the technical challenges of building and managing mobile enterprise applications.

Enterprise Mobility

Extending your enterprise business processes, data, and information away from the office entails many technical challenges.

- Device heterogeneity
- Data partitioning for mobile applications
- Data transport and application security
- Bandwidth, latency, and connectivity issues
- Managing devices, and the application life cycle

Each challenge presents complexity to the IT organization. These complexities may include, but not be limited to, for example, lack of integrated solutions, heavy upfront costs for large-scale customization requirements, and the lack of repeatable development and deployment models.

Sybase Unwired Platform meets those challenges by hiding system complexity while efficiently and flexibly handling complex mobility issues. Unwired Platform provides many opportunities for productivity gains through a well-defined life cycle, repeatable use of existing investments, and mobility knowledge.

Platform Components

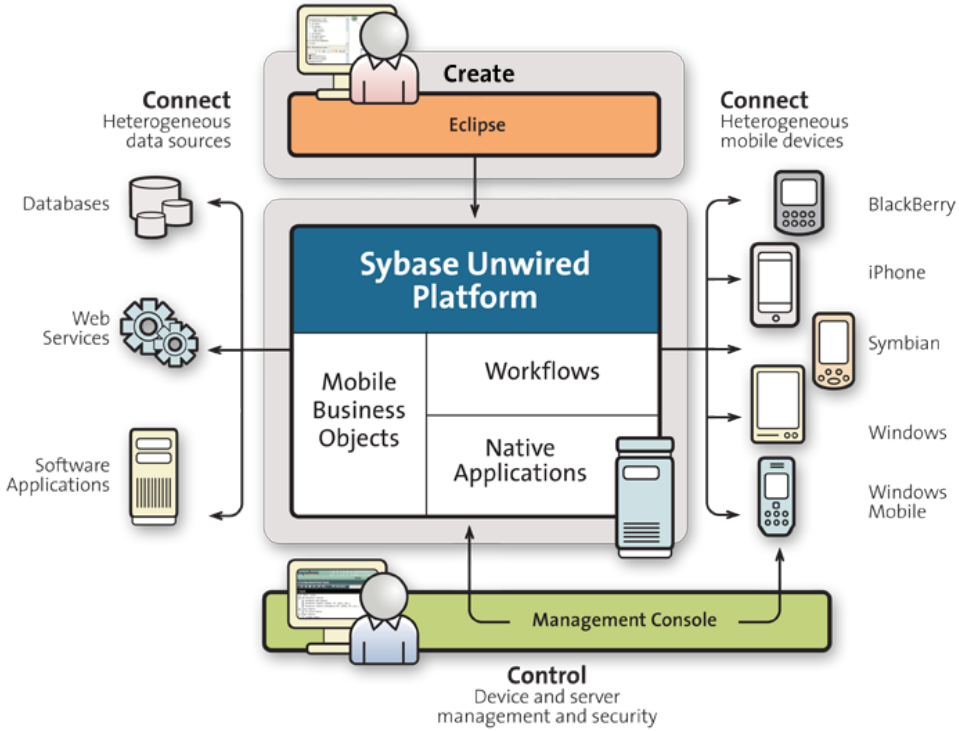
The Unwired Platform architecture consists of a server tier, a data tier, and client tier. In a production environment, the server and data tiers must be installed on 64-bit hosts.

Server-tier components integrate with back-end enterprise systems, data access and transaction services, device and application deployment, and system management functionality. The data-tier is used to store data retrieved from the backend data sources and other runtime related metadata. The client tier consists of device applications built on top of the Unwired Platform client runtime.

You can employ different secure application communication styles—replication and messaging—between the client and the server tiers. Unwired Platform uses a relay server as a component of the enterprise demilitarized zone (DMZ), to securely integrate mobile devices into your system landscape. You need not install any Unwired Platform components on the carrier network or outside your firewall; they can all be installed and controlled within your corporate networks.

For developers, Unwired Platform includes an application development environment. Unwired Platform offers a choice of application models—native application development, or simple codeless business process mobilization—by reusing your development and deployment investments in the Unwired Platform server tier. If you choose, you can install the Sybase Unwired Platform Windows Mobile .NET Component in your Microsoft Visual Studio installation. See *Application Development Environment*.

Figure 1: Sybase Unwired Platform



Core Features

Unwired Platform addresses the complex issues present in mobilizing applications to heterogeneous devices and networks.

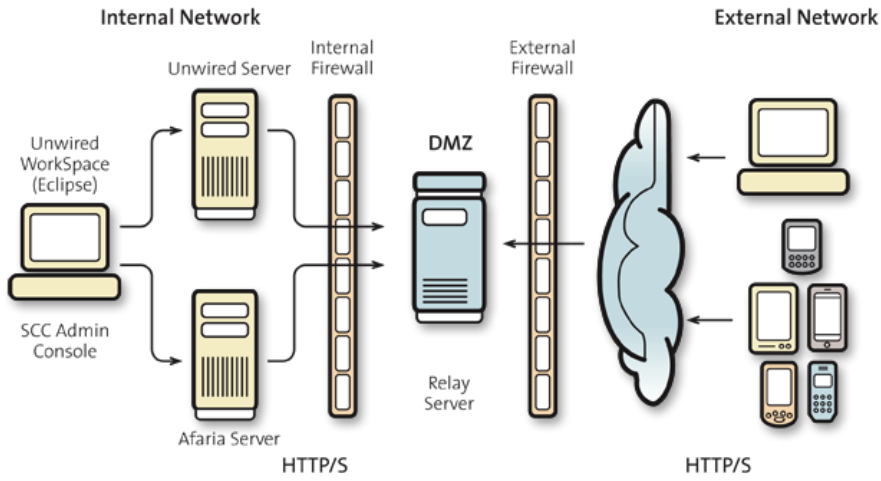
Features that address these challenges include:

- Reliable communication between devices and the server runtime to support guaranteed delivery of data and transactions, without imposing any developer or management requirements.
- Integrated support for messaging-based and replication-based synchronization paradigms for varying network and application requirements.
- Normalized data model for a variety of back-end data sources, including databases, SOAP and REST Web services, SAP® R/3® and DOE, and Remedy systems.
- Reusable, and elegant mobile business objects that capture device application data and transaction requirements.
- Integrated end-to-end security, using Secure Sockets Layer (SSL) and Transport Layer Security (TLS), for both over-the-air and data at rest.

- System security using common security providers including LDAP, Active Directory, and the operating system.
- Full development support for Eclipse development communities, and uniform development paradigm across all device platforms.
- Consistent APIs that support network-aware communication between the server and device.
- Freedom from the underlying complexity of back-end systems protocols and interaction details.
- Seamless development of applications across heterogenous device platforms.

Unwired Platform Components

The Unwired Platform configuration includes several major components.



Unwired Server

Unwired Server is an essential component of Unwired Platform, supporting data delivery to device applications by applying transactions to the consolidated database (cache), and propagating transactions to back-end systems.

Unwired Server also supports optimized access to back-end systems, messaging, security services, multitenancy capabilities, and monitoring and development activities.

Server-side APIs support advanced requirements of complex data handling, transaction execution, security customization, exception handling, and systems management

customization, all of which are intrinsic to complex corporate and multitenant-based mobile application deployment environments.

Afaria Server

Unwired Server is complemented by the device management and application deployment functionality of Afaria®.

Afaria Server is fully integrated with Unwired Server to support end-to-end management and security of devices, and over-the-air and push-based deployment of device applications.

Application Development Environment

Unwired Platform incorporates the Eclipse-based Sybase Unwired WorkSpace application development environment.

Sybase Unwired WorkSpace is an extensive set of tools that simplifies and abstracts back-end system connections, and provides a uniform object view of all enterprise data and transaction objects.

The back-end development environment also includes device application development tools that provide rapid application development and device/emulator testing support for RIM BlackBerry, Microsoft Windows Mobile, Microsoft Windows, and Apple iOS platforms.

Development is supported in a platform-neutral, as well as platform-specific, choice of languages and frameworks. For example, a Windows Mobile developer can use native C# and .NET Compact Framework programming. The same holds true for BlackBerry and iOS platforms.

The development environment provides an open, flexible platform, which enables you to integrate third-party device application components and development tools, without being locked in to a particular set of development tools. For Windows Mobile application development, you can install Sybase Unwired Platform Windows Mobile .NET Component in your Microsoft Visual Studio installation.

Development in Eclipse

Sybase Unwired WorkSpace is a plug-in to your Eclipse development environment that provides tools for creating mobile applications.

Unwired WorkSpace includes back-end integration tools that connect Unwired Server to enterprise data sources, allowing you to create mobile business objects (MBOs) from the back-end business data model.

Developers can perform MBO code generation at any time and use this MBO model code along with the user interface code that users write in a native integrated development environment (IDE). This makes the code available to transition from the Unwired WorkSpace MBO development tool to the fully extensible and open development environment provided for device platforms from third-party vendors. Optionally, use the Device Application

Designer to create prototype device applications for BlackBerry and Windows Mobile devices.

Developers can use the Mobile Workflow Forms Editor to develop message-based mobile workflow packages for Windows Mobile and iOS devices.

System Management Tools

Unwired Platform uses a Web-based administrative console to manage and administer Unwired Servers running across the corporate network.

The systems management functionality covers server configuration, command and control, device registration, multitenancy management, monitoring, device security, device management, and application deployment.

The systems management framework can integrate with Simple Network Management Protocol (SNMP) management tools to monitor specific server events. You can use administration APIs to automate and extend administration, management, and monitoring features.

Relay Server

A relay server supports load balancing and across-the-firewall deployment without opening any internal firewall ports for enterprise mobilization.

A relay server accepts and forwards requests from remote clients to Unwired Platform components. A relay server is implemented as a pair of Web extensions that run in a Web server.

Relay server supports two Web servers: IIS on Windows, and Apache on Linux. A relay server can be configured to use either HTTP or HTTPS.

Licensing Options

Sybase Unwired Platform licensing options include fixed-price and subscription-based for device clients. You can add functionality through licensing options.

Unwired Server License Options

Unwired Server licensing controls the number of devices that can access the server.

- Enterprise Developer Edition – full features supported for 20 registered devices.
- Personal Developer Edition – single-node (no clusters) features supported for five registered devices.
- Deployment Edition – full features supported (including clusters and relay server), for a specific number of devices and of CPUs.

Afaria License Option

The Afaria license option lets you provision users, devices, and applications.

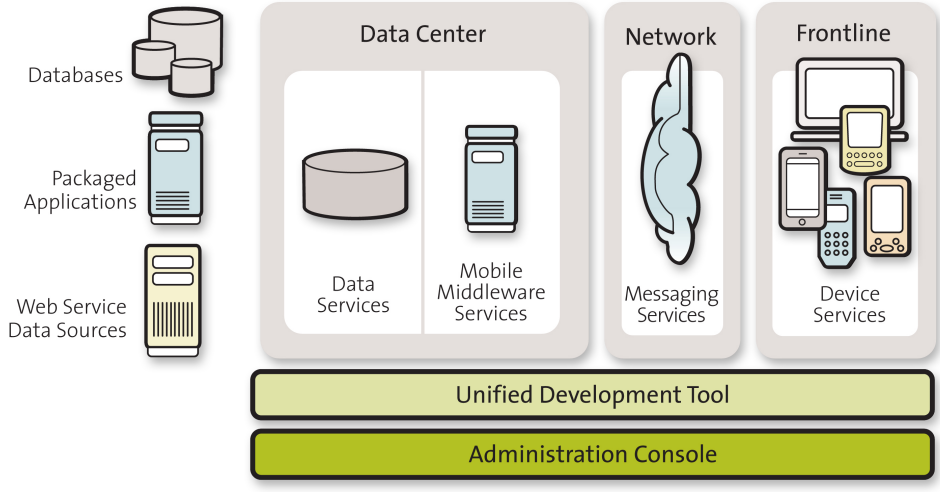
CHAPTER 2 Solution Architecture

Unwired Platform provides a solution architecture that extends the reach of your enterprise systems. The key elements of the solution architecture are mobile business objects, device applications, synchronization paradigms, security, and application deployment.

Enterprise mobile application development is often perceived as an endeavor requiring the enterprise to master complex technical details for multiple and disparate technologies. Technical challenges include:

- Connecting and interacting with various enterprise information systems and applications that exist in any large or medium size corporate IT infrastructure that supports customers, employees, sales, partners, suppliers, and executives.
- Implementing a data model that fits the application requirements and physical restrictions and capabilities of the mobile device, such as network bandwidth, storage capacity, connectivity, security, and availability of consistent data.
- Reliably propagating user transactions initiated on a device that may or may not have network connectivity.
- Ensuring secure access to data and application functionality that is available only to authorized users.

Developers need to build the most intuitive application possible without compromising on latency and the user interface. In general, once an application is built, it is passed from developer to production deployment, where the production staff manages application routing within Unwired Platform.



Managing mobile applications requires a streamlined process to deploy applications to mobile devices, and to manage routine device tasks, security, and application upgrades. Unwired Platform offers a solution architecture based on years of field-tested, enterprise-level mobility experience. The key is to manage the complexity of large scale deployment, while keeping it simple enough for line-of-business application types for small segment of users. This allows the customer base to grow, and expand their mobile reach.

The cornerstone of the solution architecture is the concept of the mobile business object (MBO). Interaction with any enterprise information system (EIS) data source requires appropriate knowledge of the objects in the EIS. The MBO represents the data model of the mobile application, and provides an abstraction layer that simplifies EIS complexity, mobile connectivity, and network issues. MBOs are built by developers familiar with the data and transactional requirements of the mobile application, and how that connects to the existing EIS data sources. The MBO development captures the intrinsic details of mobile application data and transactional functions.

The solution architecture also covers the options the platform offers for device application development for supported platforms. There are inherent issues with building applications for multiple platforms, and the details covered here help describe how Unwired Platform simplifies the most complex aspects of the device application development. Production deployment scenarios and specific platform components are discussed later.

Mobile Business Objects

Mobile business objects help form the business logic for mobile applications.

A mobile business object (MBO) is derived from a data source (such as a database server, Web service, or SAP® server). MBOs are deployed to Unwired Server, and accessed from mobile device application clients. MBOs include:

- Implementation-level details – metadata columns that include information about the data from a data source.
- Abstract-level details – attributes that correspond to instance-level properties of a programmable object in the mobile client, and map to data source output columns. Parameters correspond to synchronization parameters on the mobile client, and map to data source arguments. For example, output of a SQL SELECT query are mapped as attributes, and the arguments in the WHERE clause are mapped as synchronization parameters, so that the client can pass input to the query.

  MBO operations include parameters that map to data source input arguments. Operation parameters determine information a client passes to the enterprise information system (EIS).
- Relationships – defined between MBOs by linking attributes and parameters in one MBO, to attributes and parameters in another MBO.

You can define MBOs using either a top-down approach—first designing attributes and parameters, then binding them to a data source; or a bottom-up approach—first specifying a data source, then automatically generating attributes and parameters from it.

A mobile application package includes MBOs, roles, and data source connection mappings, and other artifacts that are delivered to the Unwired Server during package deployment.

Data and Transaction Models

Mobile business objects (MBOs) implement both a data model and a transaction model.

- Data model – attributes provide the abstract model that describes how data is represented and accessed in Unwired Platform. The goal is secure access to enterprise information system (EIS) data from the mobile application.
- Transaction model – operations provide the abstract model for data transactions. The goal is to deliver updated data from mobile devices to the EIS.

Attributes

Attributes define the structure for the data associated with the MBO instance on the mobile device.

Metadata describes the arguments that are passed to the back-end data source operations, which retrieve the data set for the MBO. Attributes correspond to the class-level properties on the mobile client object. Arguments are known as load parameters for the MBO, and can be optionally mapped to synchronization parameters for the mobile object. Default values are used when no argument values are sent from the device application, or if the argument need not be exposed to the device application. Arguments can also be bound to personalization keys to automatically populate individual user preferences.

Unwired Platform comprises both server-side and client-side components. Server-side components address the interaction between the enterprise information system (EIS) data source and the consolidated database (also referred to as the cache), while client-side components address the interaction between the consolidated database and the mobile device data store. Unwired Server brokers data synchronization and transaction processing between the server and the client.

Attributes define the scope of the device-side data store. Attributes and parameters in an MBO define the server cache. The server cache and device data store are populated by reading data from the EIS, such as a SQL select statement for a database data source.

Attributes also have additional metadata, such as specification of a synchronization parameter option that can be used to provision user-specific data. For example, a sales representative in the Eastern region may be interested only in seeing data for that region. Developers can build the application to map those preferences to MBOs to drive the data filtering specific to the user. The same data obtained from the EIS is then further partitioned and used to serve all users, thereby optimizing requests for data to the EIS and improving performance of the mobile application.

Mobility Patterns

Key data mobility patterns supported by Unwired Platform include data virtualization, publication, subscription, and operation replay.

- Virtualization – normalizes the data and semantics for interacting with different enterprise information systems (EISs), each with its own set of connection interface, data, operation, and type structures.
- Publication – data is synchronized between EIS data stores and the Unwired Server cache, to ensure that the content is consistent between systems.
- Subscription – links the client with a publication and allows the data described by the publication to be transferred to the device.
- Operation replay – supports execution of the client-initiated transactions that result in data changes on the EIS.

Operations

Mobile business objects (MBOs) may incorporate operations that change the data retrieved from the enterprise information system (EIS).

- Create, Update, Delete operations – an operation definition contains parameters that map to the arguments of the EIS operation, and can create, update, or delete data. These operations cause state change.
- Other operation – an operation definition for operations other than create, update, or delete. These operations do not cause state change.

The operation definition supports validation and error handling.

Relationships

Relationships define the data association between two MBOs by linking attributes and parameters in one MBO to attributes and parameters in another MBO.

Relationships help provision related data as one unit, and properly sequence the operations on the related MBOs based on real-time detection of the object instances used. Relationships can be one-to-many, many-to-one, or one-to-one.

From an attribute standpoint, bidirectional relationships are supported; and from an operation execution standpoint, composite relationships are supported.

Cache Groups and Synchronization Groups

The package definition can include cache groups and synchronization groups.

A cache group is a collection of mobile business objects (MBOs) that share the same cache policy, which is how the data in the Unwired Server cache is refreshed and validated for related MBOs.

A synchronization group is also a collection of MBOs that defines a unit of synchronization so that data and changes are transmitted as a unit to and from the device.

Roles

To secure access to an MBO and its operations, developers can assign logical roles, which, together with the physical role (which exists on the underlying security provider repository), helps Unwired Server perform real-time authorization checks against the user identity before allowing the request to go to the enterprise information system (EIS).

Data Sources

A data source is the enterprise information system where data is retrieved from and transactions are executed. A connection profile is a design-time connection to a data source. Connection profiles are created to specific data source by providing connection information such as host, port, login, and password among others. The connection profiles are used to define MBOs and operations, and mapped to existing, or used to create new, server connections when the package is deployed to Unwired Server..

Unwired Platform hides the interaction complexity with datasource-specific protocols, such as JDBC™ for database and SOAP for Web services.

Unwired Platform currently supports multiple EIS connection types. See *Supported Third-Party Software and Hardware* for information.

Data Caching

Data caching is initial loading (or filling) the consolidated database (CDB, or Unwired Server cache) with enterprise information system (EIS) data, then continuing to refresh the consolidated database with changes from the EIS or mobile device on an ongoing basis.

Since continual synchronization of the data between the EIS and device puts a load on the EIS, Unwired Platform provides several options for loading and refreshing the data cache.

Options include narrowing the EIS data search so that only specific data is retrieved, identifying effective policies for handling data updates once operations are performed, scheduling periodic updates to occur when system usage is low, updating only changed data in the consolidated database, and so forth.

You can use multiple options to load and refresh the right data at the right time, and to deliver the smallest, most focused payload to the mobile device.

See *System Administration* for data management information.

Data Caching Options

Data caching options include both bulk and on-demand loading schemes. There are many refinements and variations within these high-level schemes.

The primary loading schemes provide a trade-off between time and storage space.

- Bulk loading – data for all users is loaded in bulk.
- Partitioning data loading – only user relevant data, or partition, is loaded.

- On-demand loading – data is not filled, until the client performs synchronization.
- Scheduled loading – data is filled periodically according to a schedule you set.

Each option has its merits, depending on your goals. Bulk loading takes more time because data is loaded for all users, but once loaded, the data can be shared between users.

Partitioning data loading takes less time because only data the user requires is loaded. Partitioning takes more time to set up from a development standpoint, but may result in a smaller, more focused and up-to-date data load.

On demand loading delays the data loading until it is needed, similar to lazy loading, but may result in a slight delay for the user whose synchronization triggers the load.

Scheduled loading gives you control over when filling takes place.

EIS to Server Cache Integration

You can use multiple mechanisms to update the data in the Unwired Server cache.

The cache can be refreshed using the apply results cache policy (cached data corresponding to operation results is updated immediately based on data from the device, without invalidating the cache). Other cache policies include the invalidate cache policy (a restrictive policy that only updates specific cache partitions), and the no cache update policy (an operation has no effect on the consolidated database).

Developer and application administrators can also either define cache updates based on a schedule that can be set using an interval, or more granularly by setting days, dates, and specific times for refresh.

Also, for cases where the cache refresh may be expensive due to volume, you can use the data change notification (DCN) option to propagate changes from the back-end enterprise information system (EIS) to the Unwired Server or HTTP/HTTPS interface for any MBO. If you use DCN, you should not use a scheduled refresh. Choose one or the other. However, you can use an initial refresh to pull in the base set of data, and thereafter switch to DCN.

The DCN payload containing changed data is applied to the cache and the change gets published to subscribing device users based on a change detection interval time. This option is the least intrusive and most optimal for addressing high-load environments and optimizing the load on the EIS data sources to keep the Unwired Server cache consistent.

Other Key Concepts

Other key concepts for understanding mobility include object queries, synchronization parameters, result set filters, result set checkers, and personalization keys.

- Object queries – a SQL statement associated with a mobile business object (MBO) that runs on the client against data that was previously downloaded to the device. The object query returns a filtered result set (such as a single row of a table). Object queries enable discrete data handling.

- Synchronization parameters – metadata that defines how the values provided by the device application client are used to filter data, which is downloaded to the device, to provide data of interest to the user. Synchronization parameters may be used to filter the cached data, or mapped to load parameters to filter the data that is cached for the MBO and then served to the client application.
- Result-set filters – a Java API used to customize the data returned by an enterprise information system (EIS) operation. Developers can use result-set filters to alter or manipulate returned data and its structure before it is placed in the server's cache.
- Result-set checkers – a Java API that implements operation execution checks, and error handling logic for the MBO operation. Developers can use result set checkers to implement customized error checking for MBOs, since not all MBOs use a standard error reporting technique.
- Personalization keys – metadata that enables users to store their search preferences on the client, the server, or by session. The preferences narrow the focus of data retrieved by the mobile device (also known as the filtering of data between client and Unwired Server).

  Often personalization keys are used to hold backend system credentials, so that they can be propagated to the EIS. To use a personalization key for filtering, it must be mapped to a synchronization parameter.

  The developer can also define personalization keys for the application, and can use built-in personalization keys available in Unwired Server. Two key built-in personalization keys —username and password—can be used to perform single sign-on from the device application to the Unwired Server, authentication and authorization on Unwired Server, as well as connecting to the back-end EIS using the same set of credentials. The password is never saved on the server.

Object Code Generation

To access and integrate MBOs in a device application, developers have the option to generate object code for the target device platform, and then use their IDE of choice to build the native device application. The object code generation step is the bridge from the Unwired Server server-side development (MBOs) to client-side development (device applications).

The generated object code for each MBO follows a standard pattern of properties for attributes, operations, and abstractions. Object code generation is supported in the native language for each target platform. Unwired Server client libraries complement and are required for the generated object code, which together are used in the device application.

Code generation is supported for these platforms: BlackBerry (Java), iPhone (Objective-C), Windows Mobile (C#), and Windows (Java and C#). See *Sybase Unwired Platform Installation Guide* for supported versions.

Note: If developers use Sybase Unwired WorkSpace Device Application Designer for prototyping, the object code is internally generated when the device application code is generated and deployed to emulator. Additional custom coding can be done for end-to-end prototyping.

Deployment

The last step of mobile business object (MBO) development is to deploy the MBO definitions to Unwired Server as a deployment unit generated from a design-time deployment package using Sybase Unwired WorkSpace.

When you deploy MBOs to the Unwired Server, you are deploying:

- MBO definitions including attributes, operations, connections, role mappings, schedule groups, cache groups as defined in the package.
- MBO custom code related to object filters, result-set filters, and result checkers.
- Appropriate generated server-side artifacts that support the interaction with the EIS back-ends and device application.
- Other functionality captured in the MBO model.

MBOs are deployed using a deployment wizard through which you can make the choices that are appropriate for application requirements. Developers use Unwired WorkSpace to deploy a package.

The production administrator can deploy from a wizard using the web-based management console, or from the command line. Deployment-time tasks include choosing:

- Target domain – logical container for packages.
- Security configuration – used for authentication and authorization of users accessing the package.
- Role-mappings – to map logical roles to the physical roles of the back-end repository.
- Server connections mapping – to bind MBOs design-time data sources to production data sources.

Device Applications

The building blocks of the device application include the user interface (UI) layer, business logic layer, Unwired Server client object APIs for retrieving data and executing operations, frameworks for passing data and handling events, and device platform and third-party component APIs.

Unwired Platforms offers these options for developing device applications:

- Object code generation – enables you to generate object code for a device platform, then use the code in the native IDE—such as JDE for BlackBerry, and Visual Studio for Windows Mobile—to build an application. This option is suitable for developers with advanced knowledge of the target device platform development.
- Mobile Workflow Forms Editor – enables you to create a simple business workflow type of application.
- Device Application Designer – enables you to design and create prototypes for BlackBerry and Windows Mobile device platforms, using graphical device application

development tools. This method uses the MBO metadata to automatically generate the required screens to support the data viewing and operation execution of MBOs.

Application Types

Sybase Unwired Platform supports two choices for application type. First is the native application type, and the other is the container-based business workflow type.

The native application model enables the developer to write custom code (C#, Java, or Objective-C, depending on the platform) to create a device application. The native application model is supported on BlackBerry, iOS, Windows Mobile, and Windows platforms. The choice depends on the functionality desired in the application, and the need to access third-party and platform-provided APIs. Optionally, use the Device Application Designer to create prototype device applications for BlackBerry and Windows Mobile devices.

The business workflow model offers a fast and simple way to build applications that support simple business workflows, such as approvals and requests. The workflow model is supported on iOS, Windows Mobile, and Windows platforms.

Application Components

The device application components you select depend on the functionality you require in your application.

Data Access

Data access refers to both enterprise information system (EIS) data, and any local device data storage.

EIS data access requirements are fulfilled by mobile business object (MBO) definitions. This data access method is typically used in all applications.

You can also use local data access and store, which does not originate in the EIS and does not get propagated to any EIS, but is defined and captured as a local (or client-only) MBO. Local MBO objects provide object-based access and storage of application data in a device's local database.

Transactions

Transactions are the operations executed from the device application, and the processing logic for the operation. This is typically used in most applications.

The client object APIs provide access to the operation execution log, which provides support for viewing pending operations, setting unfinished activity as pending operations, reviewing operation execution status, and revisiting previously failed transactions.

Notifications

Unwired Server uses notifications to inform clients that subscribe to data whenever a change to the data is detected, or to send the data to the client database without application or user intervention.

Notification mechanisms, vary depending on the synchronization method. You can optimize notifications by fine-tuning the properties of the synchronization group and subscriptions properties.

User Interface

The user interface (UI) refers to the application screens that users use to view data and submit operations to the Unwired Server.

The user interface includes screens for viewing business data, submitting data changes or transactions, pending activities (operations), operation logs (submitted transactions), synchronization screens (to update data on-demand), and implementation logic to show or hide certain application functionality based on the user or data.

User-interface design includes choosing visualization controls, screen layout and flow design, methods of data validation, event processing and business logic, and interaction with third-party controls and APIs.

Client Object APIs

The Unwired Server client object APIs form the core building block of device applications and provide the common set of APIs for consistency across platforms, thus following the "design once and deploy anywhere" paradigm of building applications.

Client object API categories include:

- Mobile business object (MBO) object APIs – are available with each MBO and support access to MBOs attributes, operation execution, retrieving data, state information, and so forth. The APIs offer abstraction for accessing data from the EIS, sending operation executions to the EIS, updating and deleting data, and accessing persistent store in the object-oriented paradigm, among other things.
- Unwired client runtime APIs – offer consistent access to functionality across all device platforms. The APIs provide abstraction for the complex technical details of reliable communication between the device and Unwired Server. This includes a variety of network conditions for optimal transmission of data and state information, connection APIs to the back-end infrastructure, secure access to the Unwired Server data, publishing of data notifications and processing of such notifications, data protection, and access to UI management APIs and custom UI controls, such as signature control, among other things.

Synchronization Methods

Developers can use either replication-based or message-based synchronization to move data and transactions between device application clients and Unwired Server.

The choice depends on the target device platform, application requirements, target platform, and the nature of data changes and activity between Unwired Server and clients, for example, mobile workflow forms always use message-based synchronization.

Unwired Server manages and maintains data freshness between multiple data sources and device application clients through synchronization.

Replication-Based Synchronization

The replication-based synchronization model uses relational database replication, and is session-based to reduce the overhead required to maintain a continual connection. During the session, the clients submit pending operations and obtain the latest changes.

The Unwired Server can send out-of-bound device notifications to the subscribing clients; that client can then initiate a request to get the changed data.

Typically, replication-based synchronization is used in an occasionally connected or occasionally disconnected environment. Replication-based synchronization works well in situations where a large amount of data is stored on the device, continuous access to that data is required, and the data changes in bursts that require periodic synchronization between the client and server. This paradigm is supported on Windows, Windows Mobile, and BlackBerry device platforms.

Message-Based Synchronization

The message-based synchronization model is inherently asynchronous. The interaction, however, can either be synchronous, or asynchronous for messaging-based synchronization. During transmission, the client and server submit pending operations and obtain the latest changes.

Inherently asynchronous, data changes from the server to client are automatically sent and applied to client's local data store, and, similarly, operation executions from the client to server are sent in the background without requiring explicit action from device application. The developer can choose to keep transactions from being uploaded to the server by manipulating state of the submission (as pending).

Message-based services support a mobile application class that consumes message events for data and notifications or actions, and generates events to propagate changes in an always-on network availability scenario.

Message-based synchronization works well in situations that require functionality to send updates to the client intermittently with small data payload. As with replication-based

synchronization, message-based synchronization can also store large amounts of data on the device, and data exchange may be more transient, mobile workflow-related, and occurs continuously throughout the life cycle of the application. This paradigm is supported on Windows, Windows Mobile, and iOS device platforms.

Security

End-to-end security is an important component of the Unwired Platform solution architecture. Security comprises multiple components, including system security to control access to data and transactions, transport security for over-the-air and network level data protection, local data security for protecting enterprise data on the device, and device security to protect devices in case of loss or theft.

System Security

System security is built using a component-based extensible model of common security infrastructure that allows for pluggable mechanism to integrate Unwired Server with security providers including the most common LDAP providers— Active Directory, Windows OS, Remedy, and others—for authentication and authorization.

In addition, the common security infrastructure supports advanced scenarios such as authentication of back-end system interactions, and auditing of security events. Out of the box, Sybase Unwired Platform offers file-based auditing support. If a provider is not supported out of the box, a developer can write a custom provider adapter to plug in to the Unwired Server security layer.

Transport Security

End-to-end data encryption support is based on Transport Layer Security (TLS) and Secure Sockets Layer (SSL), which secures client/server communication using digital certificates and public-key cryptography.

Unwired Server synchronization protocols are supported over HTTPS (replication-based synchronization) and HTTP with proprietary encryption protocol (messaging-based synchronization).

The communication between the device client and Unwired Server can be set up through Apache or Microsoft IIS server secure ports using relay server plug-ins, which in turn communicate with Unwired Server. The connection between Unwired Server and the relay server is set up between your intranet to a DMZ server (IIS port) and thus does not require opening any internal firewall ports.

Device Data Security

Optionally you can encrypt device data so that data at rest is always encrypted and available only after users have been successfully authenticated. Unwired Client runtime APIs include an API to encrypt the device database.

Application Security

Application security is based mainly on the mapping of a mobile business object (MBO) package to a security configuration.

A security configuration defines the authentication, authorization, attribution, and auditing security provider for an application package's access control and activities. For example, for an application, an administrator may create a security configuration that points to the LDAP server for authentication and authorization, and does not associate any provider for attribution and auditing.

Authentication

Unwired Server authenticates users accessing packages using the security provider of the mapped security configuration.

Role-Based Access Control

Access to mobile business objects (MBOs) and operations that are mapped to logical roles is authorized by delegating authorization checks to mapped security provider for the package. The given user’s principal is checked for its membership in the corresponding physical role that is mapped to the logical role assigned to the MBO or operation.

Client Credential Propagation

Access to an enterprise information system (EIS) can be granted either by using a fixed set of credentials for all users, or by passing in the client user credentials.

The former is achieved by creating an Unwired Server connection, which has a fixed user and password for server connections. At deployment, the administrator maps mobile business objects (MBOs) and operations to the server connection.

Unwired Server also supports propagating the client user name and password to establish connection to the back-end using a built-in personalization key for the user name and password.

Device Security

Unwired Platform supports Afaria device management and security functionality, which includes features such as remote device locking, remote data cleanup, data fading (a feature that enables the IT administrator to lock, wipe, or reset a device that has not communicated

with the corporate network or Afaria server after a predetermined number of days), and password expiration management. Even without Afaria, the Unwired Server administrator can lock or unlock devices from accessing applications deployed to the server.

Application Deployment

During development, to test the application, the developer deploys mobile business object (MBO) packages to Unwired Server, and device application binaries and their dependencies to emulators or actual devices. Deployment to devices can be achieved by physically connecting the device to the developer's machine, and copying binaries using device-supported tools.

In a production environment, device application deployment can be achieved by using Afaria or similar third-party product. Afaria offers enterprise-class device application deployment and management features.

Device Testing

Device applications are typically tested by developers in an emulator environment. All target platforms provide emulator support from their native IDE, which enables deployment and execution of the application to the emulator.

Large-Scale Device Deployment

Some device platforms require signing the binaries before they can be deployed. For production scale deployment, application deployment is done using proven products such as Afaria.

Device Registration

Application users and their devices are registered in Unwired Server for licensing and monitoring.

Replication-based application user devices are automatically registered after the initial successful authentication with a security configuration.

Messaging-based application user devices must be registered manually. Use predefined templates to register messaging devices. Optionally, you can use Afaria to automate device registration.

System Management

The Unwired Platform management console offers integrated, Web-based access to all Unwired Servers running in the corporate network.

This enables you to view server status; perform start, stop, restart operations; configure server settings; deploy packages; configure package settings (including cache group schedule, and synchronization settings); subscriptions management (including unsubscribe, recover, suspend, and resume); push notification settings; perform role mapping; cluster management tasks; and device lifecycle and licensing tasks (including registration, unregistration, locking, unlocking, among others).

In addition, the Web console is designed for multi-tenancy in mind to support two views:

- Platform administrator view – for the administrator managing the whole environment.
- Domain administrator view – for the customers of the Unwired Server hosting provider where they can perform package management, subscription management, log viewing, role mapping and server connections management operations in one or more domains assigned to them. The domain forms the logical container and unit of separation of the artifacts contained in it. The Unwired Server administrator manages the lifecycle of the domains including users who have access to it.

Extending the operational and multi-tenancy capabilities of the platform is the monitoring component of the management console where the platform administrator can view synchronization activities, user access activities, key performance indicators for various activities in the server, cache performance, and other data being gathered. The administrator can set the monitoring configuration to enable/disable monitoring on selected domains and packages, and also perform monitoring data cleanup and export it to share with others for reporting and auditing purposes.

CHAPTER 3 Sybase Unwired Platform System Life Cycle

The "system life cycle" takes you through the process of planning and installing Unwired Platform, and integrating it with your enterprise environment; determining software deployment requirements; and setting up the Unwired Platform production environment.

The *System Administration* guide has complete information about these activities.

Planning

The planning phase includes analyzing system requirements, and creating a high-level design for your system topology and production systems.

Requirements Analysis

Analyze your requirements before designing your Sybase Unwired Platform system.

Considerations include identifying data sources, security and access policies, data concurrency, scalability and availability, number of current and projected users, device platforms, device connectivity, provisioning, and service-level agreements.

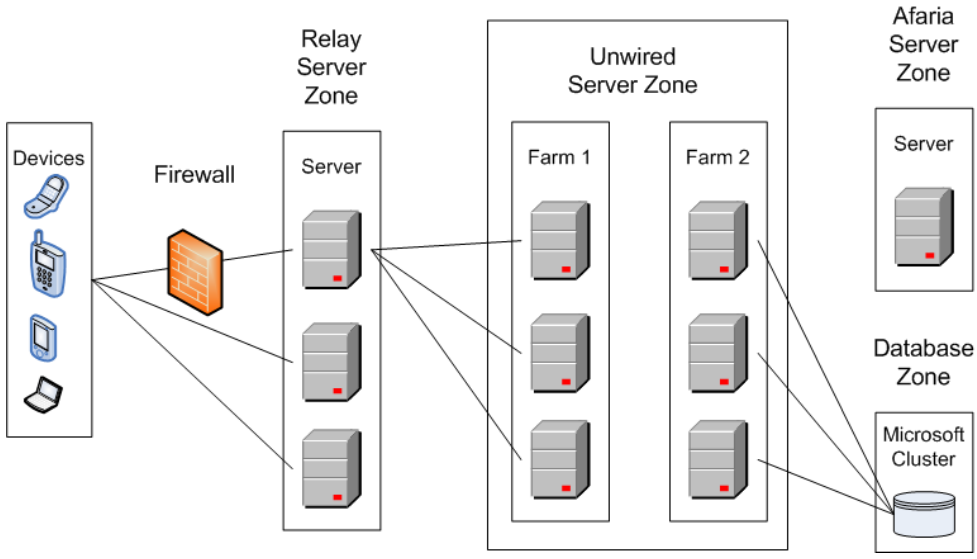
System Topology Design

Analyzing your requirements also lets you plan the system topology required to support security, failover, high availability, system load, load balancing, and database access.

Requirements information helps identify the number and specifications for machines. Sizing the system and system load enables you to determine how many Unwired Server nodes or farms are required, how many relay server nodes or farms are required, data-tier nodes, and whether Afaria nodes will be integrated.

See *Sybase Unwired Platform System Administration*.

Figure 2: Planning Sybase Unwired Platform system topology



Production Design

As part of system planning, you must also consider how to manage the production system. Some of these considerations may influence your initial ideas about system design and topology.

Considerations include device provisioning, how to manage users and devices, the manual or automated processes that are required, how to implement layers of security, and how to monitor and ensure system health.

See *Sybase Unwired Platform System Administration*.

Installing

The installation phase of the system life cycle includes installing and configuring both Sybase Unwired Platform and the system topology.

See *System Administration* and the *Installation Guide*.

Deploying Software on the Network

Deploying new or updated software—including new or updated mobile applications and mobile workflow packages, and software upgrades and patches—is an ongoing part of the system lifecycle.

See *System Administration*.

Ongoing Operations

An ongoing part of the system lifecycle is to develop operations, administration, and maintenance (OAM) processes that keeps Sybase Unwired Platform running smoothly.

- System monitoring – develop processes to monitor Sybase Unwired Platform using the Sybase Control Center, error messages, log files, and server up and down states; third-party tools; and operating system commands.
- Device provisioning – develop processes for handling existing and new users, device changes, and mobile application or mobile workflow package upgrades.
- Data subscriptions – create subscriptions, synchronization groups (messaging and replication), and device notifications (replication) to customize how updated data in the cache is delivered to the device user.
- Disaster recovery plan – develop troubleshooting processes, and solutions for routine problems. Develop a full disaster recovery plan for more serious situations.
- Backup schedule – develop a comprehensive plan for backing up the system and databases, and restoring them in case of disaster.
- Publications – develop publication subscriptions.

See *System Administration*.

CHAPTER 4 Sybase Unwired Platform Development Life Cycle

The "development life cycle" takes you through the process of designing, developing, testing, deploying and maintaining mobile business objects, device applications, and workflow packages. Understanding development life cycle helps you establish best practices for managing and monitoring Sybase Unwired Platform.

See *Sybase Unwired WorkSpace Mobile Business Object Development* and *Device Application Development*.

Designing

Design mobile business objects, the client user interface, and notification processing.

Design Requirements

Use your requirements analysis to design applications to meet your business goals.

See *Sybase Unwired WorkSpace – Mobile Business Object Development*, *Sybase Unwired WorkSpace – Device Application Development* for information to help with the design. See *Sybase Unwired Platform System Administration* for information about data management and the role of Unwired Server.

Application Type

Sybase Unwired Platform enables you to develop two basic types of applications: native applications and container-based mobile workflow packages.

If the goal of your application is to maintain the relational state between the device and the enterprise data source, and the device is occasionally connected, develop a native application using replication-based synchronization.

If the goal of your application is to keep data current between the device and Unwired Server, operations and updates can be handled as asynchronous messages, and the device is occasionally disconnected, develop a mobile workflow package, or develop a native application using message-based synchronization.

Mobile Data Requirements

Mobile data requirements include identifying the data needed and when, how current it needs to be, and which of your users can access it.

These requirements help drive update, synchronization, and access decisions. For example, if data rarely changes, synchronization is required only occasionally; if data is very volatile, a comprehensive strategy is required to ensure data is kept current on the device.

If system processing load is a concern, reduce the frequency that data is updated or accessed in the consolidated database (cache), or filter out unneeded data using system features such as SQL data queries. For data that are only used on the device to persist data across application launches, local caching or temporary usage, a local MBO can be defined.

Application Transaction Requirements

Application transaction refers to how data modifications are made on the device, and how they are propagated from the device to the data source.

These requirements determine the construction of the application. For example, you may need an application with create, read, update, and delete operations, or you may need only a subset of operations.

Device Platform Support Requirements

Device platform support requirements include identifying the device platforms on which the mobile application will run, and what common or native features can be implemented for each platform.

Device platforms may dictate the type of applications you can develop. Some features may not be available in all platforms, or may be implemented differently. The device platforms may influence the tools you use to create device applications.

Not all devices are supported for each synchronization type.

Device	Replication-based synchronization	Message-based synchronization
BlackBerry	Supported	Not supported
Windows Mobile	Supported	Supported
Windows	Supported	Supported
iPhone	Not supported	Supported

Data Source Requirements

Identify which enterprise information system (EIS) data source needed.

Consider the synchronization method, how to reduce load on both the EIS and Unwired Platform, and how to trap error messages from the EIS. You may need to write custom code to

access legacy systems, filter out unwanted data, identify specific data needed for a mobile business object, and report EIS errors.

Development Tool Requirements

Identify the development tools to use to create your mobile applications, including Sybase Unwired WorkSpace and custom coding options.

Mobile Business Object Design

Design the mobile business objects (MBOs) to connect to a data source, and to implement business logic.

A major consideration for designing MBOs is how to get data out of the enterprise information system (EIS) that is specific enough for the mobile device application or workflow package. Strategies include:

- Loading schemes to populate MBOs.
- Cache update policies to keep the MBO current.
- SQL-based object queries to filter out unneeded data.
- Application-side personalization keys to narrow the search.
- Object queries to access the local device database, as well as the back-end system, when Unwired Server is not available on demand.

More advanced strategies may implement the result-set filter and result-set checker APIs and coding to fine-tune processing.

Also consider how to perform back-end EIS operations (such as creating, editing, and deleting data), process transactions between the EIS and the mobile device, keep data up-to-date without putting undue load on system processing, and establish relationships between MBOs. You may also need to consider role mapping, logical roles, and security access in your design.

Sybase Unwired WorkSpace provides the development tools for creating MBOs that can meet these needs.

See *Sybase Unwired WorkSpace – Mobile Business Object Development* for information about developing and working with MBOs.

User Interface Design

Design user interfaces for viewing and interacting with data and operations on the mobile device.

Consider what data and how much to present on the device (within memory limitations and other device constraints), and how to create data subsets for limited resources on the device. Remember that each device platform has different controls and physical forms that determine presentation and how the user interacts with the data.

Use storyboards to plan the screen flow and decision points. Create "mockup" screen designs that specify placement of controls, logic, and validation logic. Determine what changes may be needed to adapt the design for specific device types. Identify any customization that may be

needed. You can use the Device Application Designer to design and create prototype BlackBerry and Windows Mobile device applications.

See *Sybase Unwired WorkSpace – Device Application Development* for information about tools for prototyping the user interface.

Data Consistency Design

Keep data consistent between the data source and the device.

Consider whether to refresh data on devices on a scheduled basis or on request, and the impact of the decision on system load. Strategies include:

- Cache design – design the consolidated database (cache) to refresh client application data on-demand.
- Synchronization groups – identify the mobile business objects (MBO) that are to synchronized as a group.
- Cache groups – specifies data refresh behavior for every MBO in the group.
- Data change notifications (DCNs) – define how often to update data in the cache after data changes in the source.

See *Sybase Unwired Platform – Mobile Business Object Development* for information mobile business object mobility properties (such as synchronization, load parameters, cache groups, cache update policies, object queries, and result-set filters). See *Sybase Unwired Platform System Administration* for information about data management concepts.

Developing

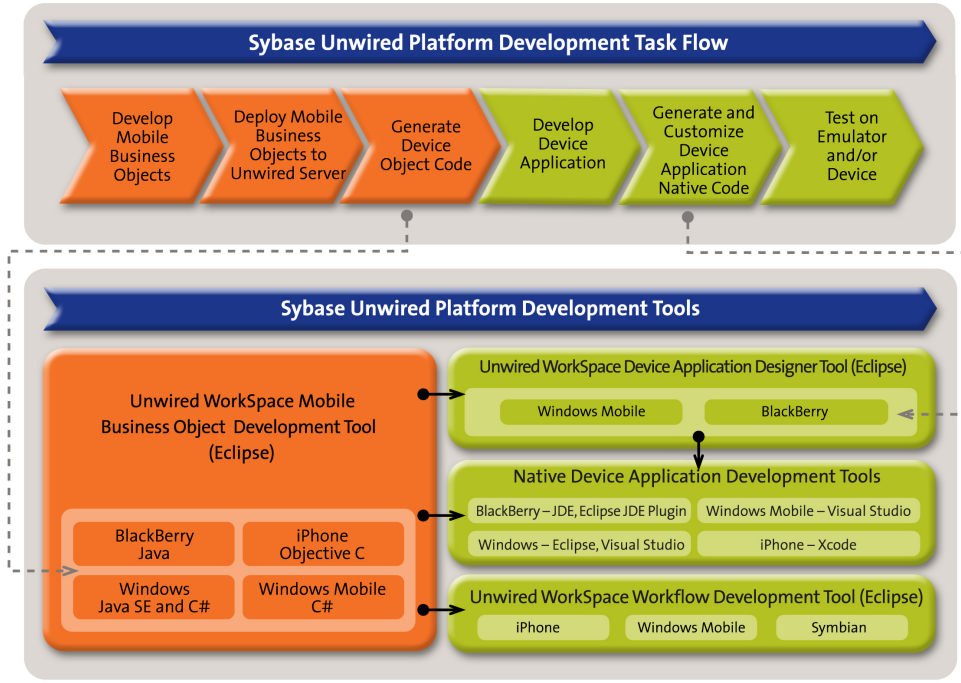
Develop mobile business objects and the device client user interface.

Development Task Flow

Describes the high-level steps to create device applications.

1. Develop a project and mobile business objects.
2. Deploy mobile business objects to Unwired Server.
3. Generate device object code.
4. Develop the device application.
5. Generate and customize device application native code.
6. Test the device application on an emulator or a physical device.

Figure 3: Device application development life cycle and tools



Mobile Business Object Development

Develop mobile business objects using Sybase Unwired WorkSpace.

Create a project, and then define one or more mobile business objects (MBOs), using the Sybase Unwired WorkSpace wizards, tools, and utilities. Creating MBOs includes connecting to a data source, setting parameters for interacting with the data, mapping data attributes, creating operations, assigning roles, and defining relationships between MBOs.

You can use replication-based synchronization or message-based synchronization for MBOs. The synchronization mode is fixed at package deployment time. A special case MBO can be used on the device to interact with a device client database. Multiple developers can collaborate on a project and do MBO development.

Once the MBOs are developed, deploy the deployment package to Unwired Server. This makes the objects ready and Unwired Server capable of servicing requests from the device application.

See *Sybase Unwired WorkSpace - Mobile Business Object Development* for information about developing and working with MBOs.

Device Application Development

Developers can use the tool best suited to develop device applications.

For the most creativity and control, you can use Sybase Unwired WorkSpace wizards to generate device object code for one or more device platforms (such as BlackBerry, Windows, Windows Mobile, and iOS). Then use the generated code as a base in a native integrated development environment (IDE), such as JDE, Eclipse JDE, Mac Xcode, and Microsoft Visual Studio, to create a customized device application. You can then use client object APIs to extend and fine-tune functionality.

For mobile workflows, you can use the Mobile Workflow Forms Editor to develop message-based, mobile workflow forms without device-side coding for Apple iPhone and iPod touch, and Windows Mobile devices.

See *Sybase Unwired WorkSpace – Device Application Development* for information about developing device applications using code generation, and about developing mobile workflows using the Mobile Workflow Forms Editor. See the *Developer References* for information about using the client object APIs on specific platforms.

See *Sybase Unwired Platform Installation Guide* for supported versions.

Testing

Test device application and workflow package on a simulator or emulator, and make adjustments before deploying to the production system.

Download emulators for devices such as Windows Mobile, and use them from the Sybase Unwired WorkSpace development environment. Alternatively, you can use Active Sync to connect the device to your development computer, and deploy and test the application on the device.

Download simulators for devices such as BlackBerry, including the MDS server, and integrate them with the Sybase Unwired WorkSpace development environment.

Deploying

Deploy mobile business objects, projects, device applications, and device clients to make them available to the system administrator to provision and manage in the production environment.

Application Deployment

Developers deploy mobile business object definitions to Unwired Server as a deployment package.

The system administrator configures and deploys the packages to a domain, and deploys application binaries to mobile devices. The system administrator manages packages depending on their type:

- Replication-based synchronization MBO packages – use an application layer service that synchronously pulls data changes from Unwired Server. Package data is stored on the device, and can be synchronized through a server-initiated push notification transmitted at defined intervals, or based upon the occurrence of a synchronization event. Data updates occur in bursts because of periodic synchronization between the client and server.

  The administrator configures cache groups, synchronization groups, and subscription templates to preset settings for push notifications. On a routine basis, the administrator checks the cache status, client log, and error history for the MBO and operations, among other things.
- Message-based synchronization MBO packages – create a device service that uses a dedicated channel that is always open to asynchronously push data changes, requests, and notifications between client and server.

  The administrator configures cache groups and synchronization groups, registers devices for application users, and configures templates or individual device settings, among other things. On a routine basis, the administrator manages subscriptions, and checks the cache status, client log, and error history for the MBO and operations, among other things.
- Mobile workflow packages – create a mobile workflow with the Mobile Workflow Forms Editor. This tool allows the developer to design mobile workflow screens that implement create, update, and delete operations, and object queries, for a mobile business object.

  The administrator configures cache group, synchronization group, and registers devices for application users, and configures templates or individual device settings, among other things. In addition, the administrator deploys mobile workflow packages and configures their parameters (such as context variables and matching rules) and assigns the workflow to users' devices. On a routine basis, the administrator checks the queue status and error history for the MBO package and workflow package.

See *Sybase Unwired Workspace – Mobile Business Object Development* for information about deploying packages; see *System Administration* for information about package administration.

Device Client Runtime Component Deployment

Some devices require a client runtime component to be deployed in addition to the device client application.

One or more of the following device runtime components would need to be deployed, depending on the choice of deployment method and the type of the application:

- Afaria clients for Windows, Windows Mobile, BlackBerry, and Symbian devices.
- Device runtime for replication-based applications (if using notifications), or messaging-based synchronization binaries for Windows, Windows Mobile, and BlackBerry devices.
- Client container for workflow packages.

No separate client runtime component is required for iPhone or Symbian devices. The deployed application includes the necessary runtime.

Device Client Application Deployment

A device client application refers to an application running on a remote device. The device client application can be deployed using Afaria or any third-party product. The client application then accesses enterprise resources remotely.

Device Provisioning and Management

When you "provision" devices it enables them to interact with the Unwired Platform environment. The provisioning process differs, depending on the device and synchronization type, and is generally managed by the system administrator.

- Device provisioning options include cradles with desktop device managers, Afaria over-the-air (OTA), and iTunes (for iPhone only).
- Devices used exclusively with replication-based synchronization (RBS) mobile business objects are automatically registered when they initially synchronize with Unwired Server.
- Devices used with message-based (MBS) synchronizations mobile business objects or workflow packages require manual device registration and activation to make the device and its associated user part of the Unwired Platform mobility system. This can be done either by the user as part of the registration process, or by the system administrator.

Optionally, you can use Afaria to extend Unwired Platform functionality by providing additional device client management features such as deploying client and runtime infrastructure components over-the-air, tracking assets, and securing devices and data.

Maintaining

Maintaining mobile business objects and device applications is an ongoing activity. Package and application versioning, and upgrading Sybase Unwired Platform and its components is part of the development life cycle.

Package Versioning

Packages deployed to Unwired Server are identified with a unique version.

Multiple versions of the same package can be running simultaneously, which means the consolidated database (or cache) includes multiple copies of the same data until all mobile applications have been switched to a new version. As a caution, there is always a consistency issue between multiple copies of the data.

Message-based synchronization uses subscription information to determine if there are any outstanding mobile applications that still subscribe to a particular version of the package.

Replication-based synchronization has no such information to reliably determine if there are still referencing mobile applications. Sybase suggests that you set up best practices for modifying and deploying new versions of mobile business objects and packages.

Application Versioning

If you have make a change to device application, you must redeploy the application binaries to the device. Such application upgrades need to be carefully planned so that currently pending operations on the device are sent to the backend before the application is upgraded.

Upgrades and Patches

Ongoing maintenance of Sybase Unwired Platform includes upgrading the software, and applying patches.

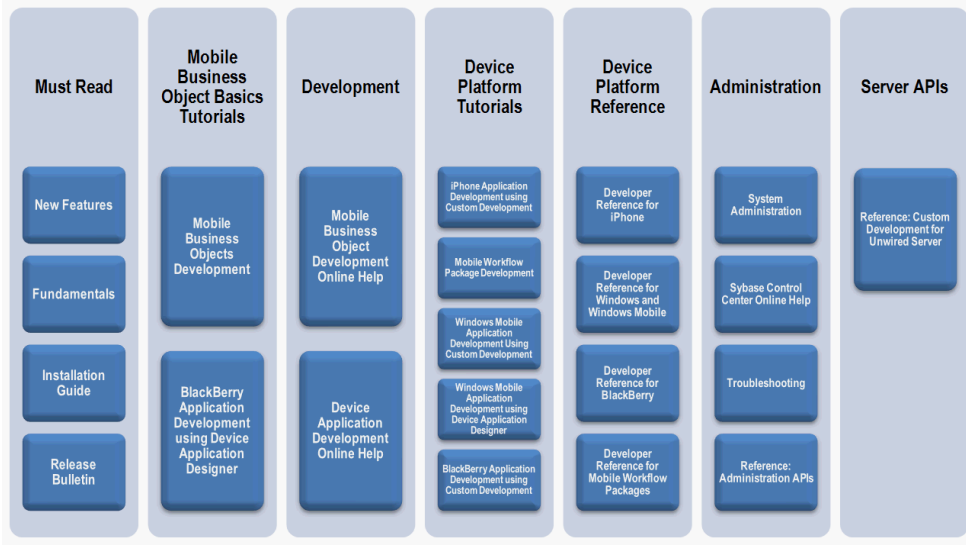
Unwired Platform is continuously under development, with new features being added and existing features being improved. Sybase recommends that you routinely apply upgrades and keep current with the latest software version.

Periodically, Sybase issues software patches to fix problems or improve performance. These patches, which help ensure a high-quality installation, are posted on the Sybase download Web site.

CHAPTER 5 Documentation Roadmap

Use the documentation roadmap to get started with Sybase® Unwired Platform.

Figure 4: Documentation roadmap



Documentation Road Map for Unwired Platform

Learn more about Sybase® Unwired Platform documentation.

Table 1. Unwired Platform documentation

Document	Description
Sybase Unwired Platform Installation Guide	Describes how to install or upgrade Sybase Unwired Platform. Check the *Sybase Unwired Platform Release Bulletin* for additional information and corrections. Audience: IT installation team, training team, system administrators involved in planning, and any user installing the system. Use: during the planning and installation phase.

Document	Description
Sybase Unwired Platform Release Bulletin	Provides information about known issues, and updates. The document is updated periodically. Audience: IT installation team, training team, system administrators involved in planning, and any user who needs up-to-date information. Use: during the planning and installation phase, and throughout the product life cycle.
New Features	Describes new or updated features. Audience: all users. Use: any time to learn what is available.
Fundamentals	Describes basic mobility concepts and how Sybase Unwired Platform enables you design mobility solutions. Audience: all users. Use: during the planning and installation phase, or any time for reference.
System Administration	Describes how to plan, configure, manage, and monitor Sybase Unwired Platform. Use with the *Sybase Control Center for Sybase Unwired Platform* online documentation. Audience: installation team, test team, system administrators responsible for managing and monitoring Sybase Unwired Platform, and for provisioning device clients. Use: during the installation phase, implementation phase, and for ongoing operation, maintenance, and administration of Sybase Unwired Platform.

Document	Description
Sybase Control Center for Sybase Unwired Platform	Describes how to use the Sybase Control Center administration console to configure, manage and monitor Sybase Unwired Platform. The online documentation is available when you launch the console (**Start > Sybase > Sybase Control Center**, and select the question mark symbol in the top right quadrant of the screen). Audience: system administrators responsible for managing and monitoring Sybase Unwired Platform, and system administrators responsible for provisioning device clients. Use: for ongoing operation, administration, and maintenance of the system.
Troubleshooting	Provides information for troubleshooting, solving, or reporting problems. Audience: IT staff responsible for keeping Sybase Unwired Platform running, developers, and system administrators. Use: during installation and implementation, development and deployment, and ongoing maintenance.

Document	Description

Getting started tutorials	Tutorials for trying out basic development functionality.
	Audience: new developers, or any interested user.
	Use: after installation.
	• Learn mobile business object (MBO) basics, and create a mobile device application:
	  • *Tutorial: Mobile Business Object Development*
	  • *Tutorial: BlackBerry Application Development using Device Application Designer*
	  • *Tutorial: Windows Mobile Device Application Development using Device Application Designer*
	• Create native mobile device applications:
	  • *Tutorial: BlackBerry Application Development using Custom Development*
	  • *Tutorial: iPhone Application Development using Custom Development*
	  • *Tutorial: Windows Mobile Application Development using Custom Development*
	• Create a mobile workflow package:
	  • *Tutorial: Mobile Workflow Package Development*

Sybase Unwired WorkSpace – Mobile Business Object Development	Online help for developing MBOs. Audience: new and experienced developers. Use: after system installation.
Sybase Unwired WorkSpace – Device Application Development	Online help for developing device applications. Audience: new and experienced developers. Use: after system installation.

Document	Description
Developer references for device application customization	Information for client-side custom coding using the Client Object API. Audience: experienced developers. Use: to custom code client-side applications. • *Developer Reference for BlackBerry* • *Developer Reference for iOS* • *Developer Reference for Mobile Workflow Packages* • *Developer Reference for Windows and Windows Mobile*
Developer reference for Unwired Server side customization – *Reference: Custom Development for Unwired Server*	Information for custom coding using the Server API. Audience: experienced developers. Use: to customize and automate server-side implementations for device applications, and administration, such as data handling. Dependencies: Use with *Fundamentals* and *Sybase Unwired WorkSpace – Mobile Business Object Development*.
Developer reference for system administration customization – *Reference: Administration APIs*	Information for custom coding using administration APIs. Audience: experienced developers. Use: to customize and automate administration at a coding level. Dependencies: Use with *Fundamentals* and *System Administration*.

Index

A

C

D

E

F

G

K

L

M

O

P

R

S

T

U

V